Reporte de práctica de laboratorio

***Utilizar un cliente de control de versiones***

31 marzo 2019

Maestría en Sistemas Computacionales

*Aplicaciones y Servicios en la Nube*

Prof. Mtro. Rodolfo Luthe Ríos

Ing. César Alan López López

[ms711016@iteso.mx](mailto:ms711016@iteso.mx)

# Introducción

Utilizar un cliente de control de versiones, Configurar servicios de control de versiones centralizados y Controlar las versiones de un documento

Actividades

• Configurar un repositorio local

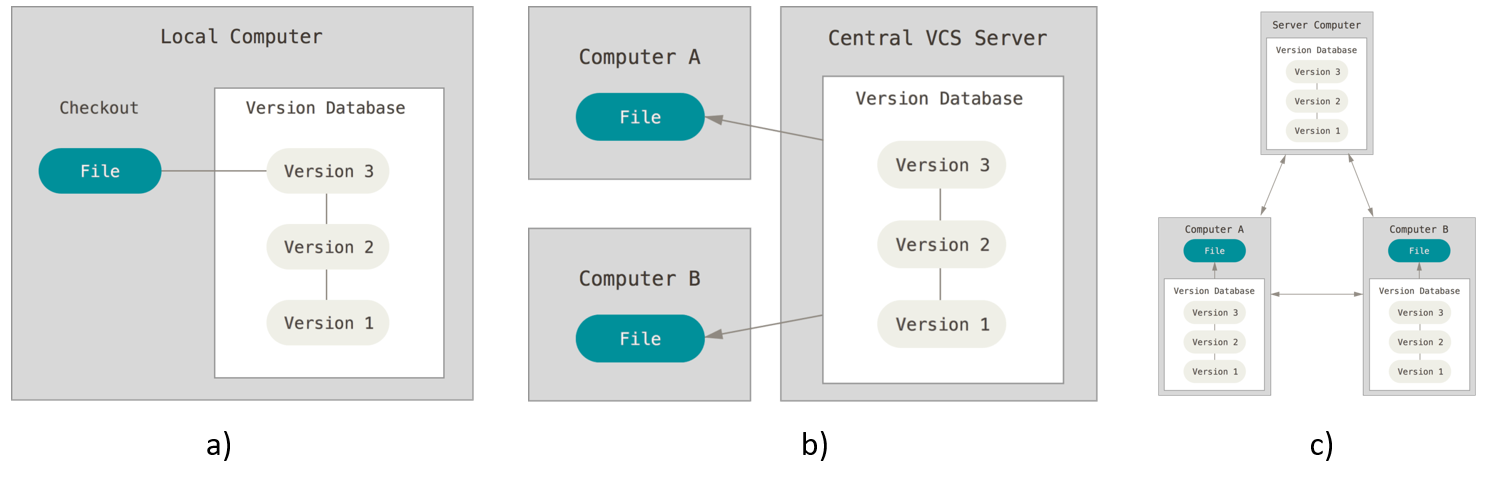
• Crear un repositorio en GitHub

• Crear un reporsitorio en CodeCommit

• Controlar las versiones de un documento

Marco Teórico

Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante. Un sistema de control de versiones (VCS por sus siglas en inglés) te permite regresar a versiones anteriores de tus archivos, regresar a una versión anterior del proyecto completo, comparar cambios a lo largo del tiempo, ver quién modificó por última vez algo que pueda estar causando problemas, ver quién introdujo un problema y cuándo, y mucho más. En la siguiente imagen se muestran las principales arquitecturas de VCS.



### Sistemas de Control de Versiones Locales

Un método de control de versiones, usado por muchas personas, es copiar los archivos a otro directorio. Este método es muy común porque es muy sencillo, pero también es tremendamente propenso a errores. Es fácil olvidar en qué directorio te encuentras y guardar accidentalmente en el archivo equivocado o sobrescribir archivos que no querías.

### Sistemas de Control de Versiones Centralizados

Estos sistemas, tienen un único servidor que contiene todos los archivos versionados y varios clientes que descargan los archivos desde ese lugar central. Las personas saben hasta cierto punto en qué están trabajando los otros colaboradores del proyecto. Los administradores tienen control y es mucho más fácil administrar. Sin embargo, esta configuración también tiene serias desventajas. Al ser un servidor centralizado. Si ese servidor se cae, entonces durante nadie podrá colaborar o guardar cambios en archivos en los que hayan estado trabajando. Si el disco duro en el que se encuentra la base de datos central se corrompe, y no se han realizado copias de seguridad adecuadamente, se perderá toda la información del Proyecto.

### Sistemas de Control de Versiones Distribuidos

En los sistemas de Control de Versiones Distribuidos (DVCS) los clientes no solo descargan la última copia instantánea de los archivos, sino que se replica completamente el repositorio. De esta manera, si un servidor deja de funcionar y estos sistemas estaban colaborando a través de él, cualquiera de los repositorios disponibles en los clientes puede ser copiado al servidor con el fin de restaurarlo. Cada clon es realmente una copia completa de todos los datos. Se encargan de manejar numerosos repositorios remotos en los que puedes colaborar simultáneamente con diferentes grupos de personas en distintas maneras dentro del mismo proyecto. Esto permite establecer varios flujos de trabajo que no son posibles en sistemas centralizados, como pueden ser los modelos jerárquicos.

## Clientes de control de versiones

Git es un sistema de control de versiones (VCS) de código abierto distribuido que le permite almacenar código, rastrear el historial de revisiones, fusionar cambios de código y volver a la versión de código anterior cuando sea necesario.

* Seguimiento de cambios históricos
* Trabajar en equipo
* Mejore la velocidad y la productividad del equipo
* Disponibilidad y redundancia
* Git es el estándar de la industria

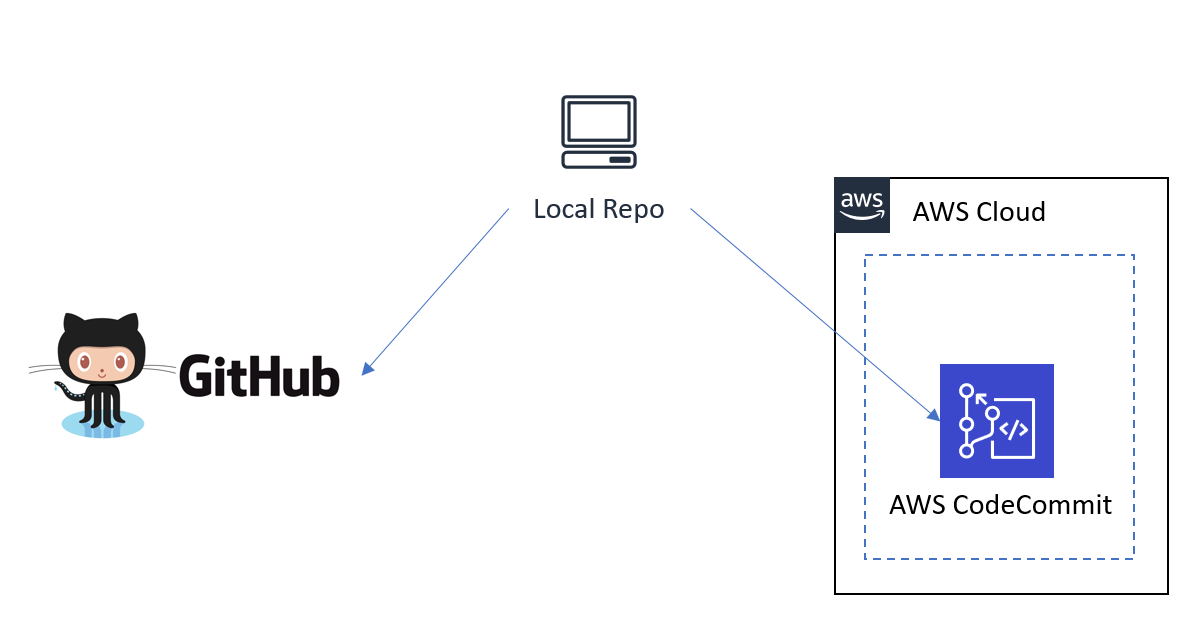
AWS CodeCommit es un servicio de control de versiones alojado en Amazon Web Services que puede utilizar para almacenar y administrar recursos de forma privada (como documentos, código fuente y archivos binarios) en la nube. Sus principals características son:

* Basado en Git, admite comandos de Git, además de comandos de AWS CLI y API.
* Ventajas de un servicio totalmente administrado alojado por AWS.
* Los repositorios de CodeCommit están cifrados en reposo y en tránsito.
* Permite trabajar de forma colaborativa en el Código.
* Escalar fácilmente proyectos de control de versiones.
* No hay límite de tamaño ni de tipos de archivos que se pueden almacenar.
* Se integra con otros servicios de AWS y de terceros.
* Puede migrar a CodeCommit desde cualquier repositorio basado en Git.

Diferencia de CodeCommit con el versionamiento de S3

El control de versiones de Amazon S3 admite la recuperación de versiones antiguas de archivos, pero no se centra en las características de seguimiento de archivos en colaboración que necesitan los equipos de desarrollo de software, CodeCommit sí permite eso.

# Diagrama



## Configurar repositorio local

1. Instalar el cliente de git en una maquina

a. https://git-scm.com

2. Configurar git con su cuenta de iteso

a. git config --global user.name "John Doe"

b. git config --global user.email johndoe@example.com

3. Configurar el repositorio

a. Crear en Mis Documentos una carpeta llamada git

b. Desde consola cambiarse a ese directorio (o en Windows click derecho sobre la

carpeta y “Git Bash here”)

i. git init

4. Control de versiones

a. Crear en el directorio git un archivo: versiones.txt con el contenido: Versión 1

b. Añadir el archivo al seguimiento de git, desde la consola en la carpeta: git add .

i. Es importante el “.” para incluir todos los archivos en el tracking

c. Hacer un commit de los cambios: git commit –poner mensaje versión inicial

d. Editar el archivo y cambiar el contenido a: Versión 2

i. Hacer add y commit de los cambios:

1. git add .

2. git commit -m “segunda version”

e. Hacer lo mismo para la Versión 3

f. Consultar historial de vesiones: git log

## Configurar repositorio en GitHub

1. Crear una cuenta con su correo de iteso en GitHub

2. Crear un repositorio en GitHub

3. Consultar el URL del repositorio

4. Añadir al repositorio local el repostorio de GitHub, desde la carpeta de git:

a. git remote add Hub <URL>

b. git push Hub master

5. Refrescar la pantalla del repositorio en GitHub ¿Qué cambio observan?

## Configurar repositorio en AWS CodeCommit

1. Crear credenciales para CodeCommit

a. Desde la consola de IAM y seleccionar el usuario deseado

b. En Security Credentials generar para: HTTPS para CodeCommit

2. Desde la consola de CodeCommit crear un repositorio

3. Consultar el URL del repositorio

4. Añadir al repositorio local el repostorio de CodeCommit, desde la carpeta de git:

a. git remote add AWS <URL>

b. git push AWS master

5. Consultar el repositorio de CodeCommit

## Controlar las versiones del entregable de la práctica 10

1. Crear el documento del reporte, y añadirlo para tracking

2. Hacer add y commit al terminar cada sección del reporte -Marco, Diagrama, etc- con el comentario correspondiente

3. Mostrar log de los cambios al documento

4. Subir PDF del reporte a Moodle, compartir el URL del documento en CodeCommit

# Problemas y Soluciones

En esta práctica no se encontraron problemas

# Experimentos y Resultados.

# Costo

Conclusiones

# Bibliografía